Assignment 2
Ray Tracing – 1
Due: September 29th, 2010

Overview:

In this assignment, you will create a simple ray-tracing application capable of rendering scenes consisting of spheres lit by a single point light source. Scenes will be in an XML specification. This is part 1 of a 2 part assignment. In the second part you will extend this raytracer to incorporate triangle meshes, shadows, and other extensions for more realistic images.

Getting Started:

To facilitate scene loading, you can find an example XML parser and scene representation code here, there is code in both C++ and Java.

Online is posted the XML scene description, example scenes, and example renderings.

At this point, you will have experience with GLUT and EasyBMP. These will give you 2 options for displaying your outputs.

Submission Instructions:

Create a sample webpage with:

-Your source code
-Output images
-You must show *at least* one of the sample scenes and one new scene of your own
-Brief description of your implementation, any issues you saw, or any extra credit
-Submission to Art Contest (optional)

Assignment Requirements:

As this is a 2 part assignment, part 1 will be graded out of 50 pts. The number is front is how many points the feature is worth. Partial credit will be given to features that "sort of" work. The required minimum feature set is underlined.

Scene Setup:

- *(5) Camera placement, film resolution, aspect ratio*
- *(5) User specified background colors*
- (5) UI + OpenGL output
- (5) BMP output

Primitives:

- *(5) Spheres*
- (5) Difference/Intersection of spheres (Constructive Solid Geometry)

Lighting:

- *(5) Ambient lights*
- *(5) Point light sources*
- *(10) Phong lighting*
- (5) Directional light sources
- (5) Spot light sources

Sampling:

- *(5) Basic Sampling*
- (5) Jittered Supersampling
- (5) Adaptive Supersampling
- (5) Motion Blur
- (5) Depth of Field

Materials:

- *(5) Color & Specularity*
- (5) Refraction
- (5) Reflection

Hints:

-Test your program incrementally

-Try hard to test incrementally

-Honestly, incremental testing will really help!

-By testing incrementally I mean, you should get some very small thing working first then move on to more complex features. First generate a black image of the correct dimensions. Then create a scene with just a sphere. Have your raytracer return white whenever it hits the sphere. You should see a white sphere on a black background. Now make sure you feel confident in your ray/sphere intersection code. Move the sphere around, make sure the right results happen. When this works, mess around with the image aspect ratio (make it very narrow), make sure everything works as expected. Then instead of coloring the sphere white use the real material color. Then add support for point lights, then phong shading. Only move on to a new feature when you're sure everything else works so far.

-If you don't do this, you'll end up with a big pile of broken code and I won't be able to help you.

-Test incrementally

-Please, please test!